Traveler
Software Requirements Specification

Aliya Ware, Tammas, Thomas, Jordan, Evan
https://github.com/ScriptkidHicks/Traveler

Table of Contents:

1. The Concept of Operations (ConOps)

The concept of operations (ConOps) “describes system characteristics for a proposed system from the users’ viewpoint.” (IEEE Std 1362-1998) The ConOps document communicates overall system characteristics to all stakeholders.

1.1. Current System or Situation

Making sure people get their deliveries on time is a very stressful thing for many companies. Delivery drivers need to be able to choose the quickest and most efficient route to get to all of their destinations and back home on time. Most delivery systems do not have an efficient way to calculate the quickest route to each destination, keeping them stressed out about getting their jobs done on time. These businesses are in need of a service that can ease this stress off of them and produce an efficient delivery route for them.

1.2. Justification for a New System

As consumption increases, and especially online consumption due to covid, delivery companies are put under an incredible amount of stress to find routes for drivers in order to keep up with their deliveries. At this point, drivers barley have time for a bathroom or snack break because this stop would critically set their delivery schedule back. As deliveries are delayed because driers are still using their old route calculating services, packages have been arriving later than ever. Though in the past when demand was not as high, drivers had less stress about getting deliveries to people on time because their route calculating methods were working totally fine. However, it is crucial now more than ever that they are taking the most optimal route to hit all of their destinations as quickly as possible.

Our web-application provides the best solution to this problem. We focus on producing a route for delivery drivers that hit all of their destinations in the shortest amount of time. This increases consumer happiness and satisfaction with the shipping service, and the delivery driver less stressed. The routes our automated software system can provide will cut down time on the road (gas used), miles driven, and the hours and stress of planning what route to take each day.

1.3. Operational Features of the Proposed System

Our new system has the newest and most improved technology that provides drivers with the the most optimal route they should take to reach all of their destinations. The driver will enter in all the destinations to which they need to get to in the day, and our route calculator will optimize the order and path the driver should take to complete their deliveries. The routes can be different every day, however our software will optimize any set of delivery routes that need to be taken that day. While other systems needed planning of where to stop first and would then optimize routes to each stop, our system will optimize the destination and route path.

1.4. User Classes

For the proposed application, Traveler, there will be one major user class: people who are driving. The system calculates the optimal route of and to destinations, but it needs the user to be able to input their start location as well as the other locations to which they need to get to. We’re assuming that the driver knows their destinations. This will only be used for optimal driving routes, not biking, bussing, walking, or any other form of transportation.

1.5. Modes of Operation

There will be only one mode for all users. There is no reason for other modes, as our software system calculates the optimal distance to travel to several destinations and back. Anyone can use it whether that be the business, the drivers, or any other clientele. They will have a profile that they sign up for or login to which will keep their destination information safe and invisible to other people.

1.6. Operational Scenarios (aka "Use Cases")

"An operational scenario [also known as "Use Cases"] is a step-by-step description of how the proposed system should operate and interact with its users and its external interfaces under a given set of circumstances. Scenarios should be described in a manner that will allow readers to walk through them and gain an understanding of how all the various parts of the proposed system function and interact. The scenarios tie together all parts of the system, the users, and other entities by describing how they interact." (IEEE Std 1362-1998)

'Operational scenarios should describe operational sequences that illustrate the roles of the system, its interactions with users, and interactions with other systems. Operational scenarios should ideally be described for all operational modes and all classes of users identified for the proposed system. Each scenario should include events, actions, stimuli, information, and interactions as appropriate to provide a comprehensive understanding of the operational aspects of the proposed system. Prototypes, storyboards, and other media, such as video or hypermedia presentations, may be used to provide part of this information.' (IEEE Std 1362-1998)

Include, in each operational-scenario (or use-case), a one-sentence description of the scenario, a list of the users or "actors" (from the user classes) involved in the scenario, the preconditions for starting the scenario, and the postconditions (the relevant status of the system and world) after the scenario is completed.

Structure the writing of operational scenarios, or use cases, so that they are easy to read, with headings and numbered steps rather than in paragraph form. For example, note how the following is well-structured to make it easy to read.

Use Case 1: Create an account.

Brief description: This use case describes how a business or driver would make an account to start using the software system.

Actors: A driver.

Preconditions:

1. The driver has access to a desktop or mobile device which has some sort of browser such as chrome or safari to access and use the system.
2. The driver has internet or cellular data access.
3. The driver has an email address.

Steps to Complete the Task:

1. Follows the prompt on the screen and enters in information in appropriate sections.
2. The Driver enters:
   - (a) Their email address
   - (b) Their name
   - (c) A password
3. The user clicks the final “create an account” button.

Postconditions:

The driver now has a secure account from which now they can continue and start using the program. They can interact with the homepage and move to the main page where they are prompted to enter in their starting address and the destinations to which they need to go.

Use Case 2: Route Calculation

Brief description: This use case describes how a driver would proceed to get their map of the optimized route.

Actors: The Driver

Preconditions:

1. The driver has 1 or more deliveries to make.
2. The driver knows where they are (starting point) and where they need to go (destination).
3. The driver can read a map.
4. The driver has internet or cellular data to be able to access map on the road
5. The driver is logged in.

Steps to Complete the Task:

1. The user will get to the main page by clicking the start button
2. The user then will be prompted to enter in their starting location in the top entry box
3. The user will enter the destinations to which they would like to travel in the boxes following.

3. The driver will then click the “calculate route” button which will take them to the results page.

Postconditions:

The driver will be able to see the optimized route and the directions to each destination.

Note that diagrams can assist with communication.

“Graphical tools should be used wherever possible, especially since ConOps documents should be understandable by several different types of readers. Useful graphical tools include, but are not limited to, work breakdown structures (WBS), N2 charts, sequence or activity charts, functional flow block diagrams, structure charts, allocation charts, data flow diagrams (DFD), object diagrams, context diagrams, storyboards, and entity-relationship diagrams." (IEEE Std 1362-1998).

2. Specific Requirements

This is where the actual requirements are specified. A requirement is a description of a behavior or property that a computer program must have, independent of how that behavior or property is achieved. Requirements must be complete, unambiguous, consistent, and objectively verifiable (see van Vliet, 2008, pp. 241-242, for a discussion of these terms). Requirements describe what the system will do, but do not commit to specific *design* details of how the system will do it.

Requirements should be organized in a hierarchy. A good organization (a) makes requirements easier to read and understand because related requirements will be near each other in the document, (b) makes requirements easier to modify and update, and (c) makes it easier to find a specific requirement. There are a number of ways that requirements can be organized to help achieve these goals.

The following section headings provide one way to organize the requirements. You can adapt this organization. For example, sections 2.1, 2.2, 2.3, and 2.4 (see headings below) describe "behavioral requirements" (Faulk, 2013). If a system supports two major user activities, it might be best to describe the behavioral requirements for each activity separately. For example, in a digital deejaying system, the two major activities could be (a) load songs into the system and (b) use the system to play songs. It might be best to fully specify everything in sections 2.1 through 2.4 first for the song-loading activity, and then for the song-playing activity.

Though there is a tradition of distinguishing between "functional" and "non-functional" requirements, with the former describing services provided by the system, and the latter describing constraints on the system and its development, in actual practice this is not a very useful distinction, and is not a good basis for structuring requirements (Faulk, 2013).

Requirements should be prioritized, with each classified as (a) must have, (b) should have, (c) could have, and (d) won't have. These can be recalled with the memory aid of MoSCoW (vanVliet 2008, p.237). When reading requirements, it should be very easy to see how each requirement is classified, such as by having them grouped by priority.

Throughout the document, lists and sublists of requirements should be indented and numbered to make it easy to read and reference the specification details. Such as:

1. *General Requirement*
    1.1. *Specific Requirement*
        1.1.1 *Requirement Detail*

Note how this permits reference to "SRS Item 1.1.1".

2.1. External Interfaces (Inputs and Outputs)

2.1.1. Creating an account

1. Description of purpose: This allows for drivers to have an account that protects the route they are traveling on and make it only visible to them.
2. Source of input/destination of output: Input will be the user's personal information such as name and email, to create the account. The output will be a successfully created account to which they can log into.
3. Valid ranges of inputs and outputs: The valid input range will be a new name and email address for each account created. If an email matches an existing account it will not make a new one because that means the email is already associated with the account. A valid output will be a single, unique account for each user. There can be no duplicate accounts.
4. Units of measure: None
5. Data formats: The account data will be concisely formatted in the top right corner of the screen showing the user's name

2.1.2. Entering addresses and getting a map

1. Description of purpose: To get an optimized route that gets the user to all of their destinations and home in the shortest amount of time
2. Source of input/destination of output: Input will be the user's starting address, followed by the destinations to which they would like to go. The output will be a map that shows the user the order in which to travel to their destinations and home. In addition to showing the optimal order of destinations, the map will also have the optimal path to each destination.
3. Valid ranges of inputs and outputs: A valid input will be proper addresses that exist on google maps. The addresses also need to be within driving range of each other. A valid address for example will not reside in the middle of the ocean or a body of water. A valid output will be an optimized route from the starting point to all of the destinations and back.
4. Units of Measure: ?
5. Data Formats: The data will be formatted in a list order. The final route will be on a map and to the left of that will be a list of addresses from starting point to each destination in the order in which they should travel to each point.

2.2. Functions

1. Validity checks on the inputs: The program will check that the user is using their same account and not creating a new one with the same email. The program will also check

that when the user is logged in and ready to input their starting and ending locations, that the addresses they are entering are valid.

2. Sequence of operations in processing inputs: Once the user has an account and is signed in, they can start listing the destinations to which they need to travel. Without an account, the user will not be able to access the route calculation page. The user can enter in valid addresses and click the "submit locations button". If the addresses are not valid, then it is not possible to provide a map from the starting location to the destinations.
3. Responses to abnormal situations, including error handling and recovery: Upon account creation, an error will pop up if the same email or username has been used before, with a prompt asking for different information. Upon sign in, if the information or account does not exist or if the information is invalid, an error message will pop up to let the user know that the email or password is invalid/does not exist. Upon address entry, if the addresses are invalid, a message will let the user know that the address does not exist. There is an autocomplete feature that will give users suggestions as to what their address may be as they are typing.
4. **Relationship of outputs to inputs, including**
    - **(a) input/output sequences**
    - **(b) formulas for input to output conversion**

2.3. Usability Requirements

Define usability requirements and objectives for the software system, include measurable effectiveness, efficiency, and satisfaction criteria in specific contexts of use. (ISO/IEC/IEEE 29148:2011)
Traveler will require that the user set up an account or log in to an existing one in order to be able to enter in their destination points. The addresses they enter must be valid points on a map. The accuracy of the map will depend on if the user entered in the proper addresses. No matter what addresses are entered, as long as they are valid and of drivable distance from each other, the resulting map will show the most optimal route.

2.4. Performance Requirements

Specify the static and the dynamic numerical requirements placed on the software or on human interaction with the software. For example: (a) Static numerical requirements may include the amount and type of information to be handled. (b) Dynamic numerical requirements may include the amount of data to be processed within certain time periods.

Performance requirements should be stated in measurable terms. For example,
"95% of the transactions shall be processed in less than 1 second"
rather than
"An operator shall not have to wait for the transaction to complete."
(ISO/IEC/IEEE 29148:2011)

Our program will only accept valid address inputs and there must be a starting address and between one and ten additional destinations for our program to calculate the optimal route.

If the user has 10 destinations they want to get to, which is the maximum number of destinations our program accepts, the route will be calculated in about .2 seconds using an intel core i7. The less destinations the quicker the program will find the optimal route, however the user should experience little to no difference in how long they wait for results.

2.5. Software System Attributes

Specify the required attributes of the software product, such as reliability, security, privacy, maintainability, or portability. (ISO/IEC/IEEE 29148:2011) Review a comprehensive list of software attributes, or software qualities, such as are provided in van Vliet (2008) Chapter 6. Decide on a relatively small number of attributes that are most important for this system. Explain why each attribute is important, and what steps or plan will be taken to achieve those attributes. This could include constraints on attributes of the system's static construction, such as testability, changeability, maintainability, and reusability. (Faulk, 2013)

This software must protect the privacy of the users starting location as well as the destinations to which they would like to go. Their account information such as username, email, and password should also be secure, making sure heir destinations protected from outside parties. The user can choose their own starting and destination addresses.

The software must be reliable so that the user can get a very accurate route in a timely manner. It must not break so that if a driver is in a hurry, they can get their route and be on their way instead of spending time figuring out what went wrong.

Traveler wants to make sure that our customers keep using our software and be able to adapt it to their needs, so we make it maintainable so that if we need to update security or route results, we can do it quickly and efficiently.

3. References

This section lists the sources cited in the creation of this template document. An SRS should reference all of the sources that it draws from. If sufficient citations are provided "in line" (at the point of reference) in the document, this section may not be necessary.

IEEE Std 1362-1998 (R2007). (2007). IEEE Guide for Information Technology–System Definition–Concept of Operations (ConOps) Document. https://ieeexplore.ieee.org/document/761853

IEEE Std 830-1998. (2007). IEEE Recommended Practice for Software Requirements Specifications. https://ieeexplore.ieee.org/document/720574

ISO/IEC/IEEE Intl Std 29148:2011. (2011). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/6146379

ISO/IEC/IEEE Intl Std 29148:2018. (2018). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/8559686

Faulk, Stuart. (2013). *Understanding Software Requirements*. https://projects.cecs.pdx.edu/attachments/download/904/Faulk_SoftwareRequirements_v4.pdf

Oracle. (2007). White Paper on "Getting Started With Use Case Modeling". Available at: https://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf

van Vliet, Hans. (2008). *Software Engineering: Principles and Practice*, 3nd edition, John Wiley & Sons.

Work Breakdown Structures. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Work_breakdown_structure.

N2 Charts. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/N2_chart.

Functional Flow Block Diagrams. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Functional_flow_block_diagram.

Structure Chart. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Structure_chart.

Data-flow Diagram. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Data-flow_diagram.

Object Diagram. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Object_diagram.

System Context Diagram. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/System_context_diagram.

Storyboard. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Storyboard.

Entity Relationship Model. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model.

4. Acknowledgements

All sources used in the creation of the document and support you received from anyone not in your team should be listed here.